Computação em Nuvem

GRADUAÇÃO EM REDES DE COMPUTADORES E
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Prof. Guto Muniz

Contêineres, Docker

GRADUAÇÃO EM REDES DE COMPUTADORES E
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

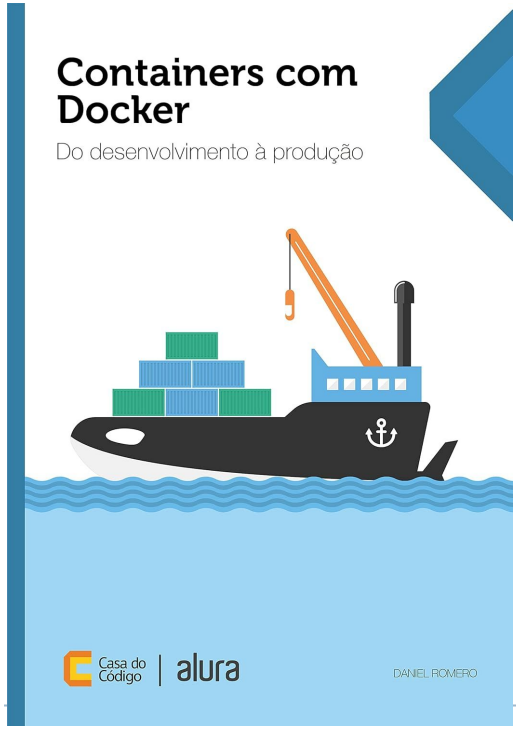
Prof. Guto Muniz

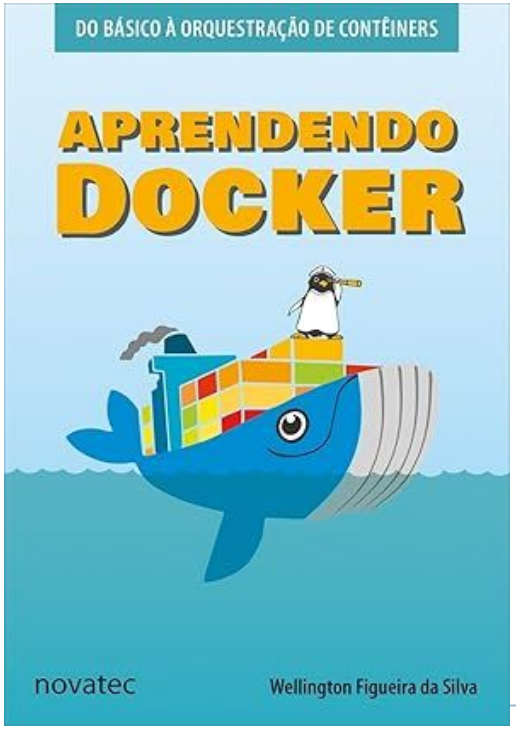
Introdução

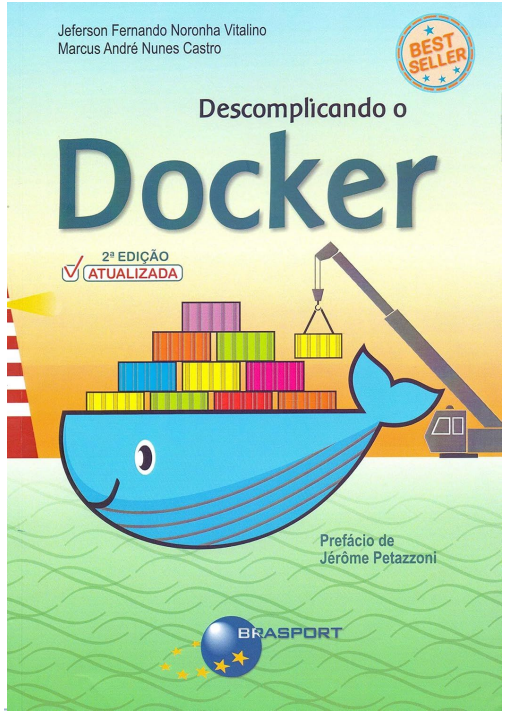
- Contêiner
- Docker

Introdução

- **Literatura**







Como funcionava?

Como funcionava?

- **Muito tempo ocioso**

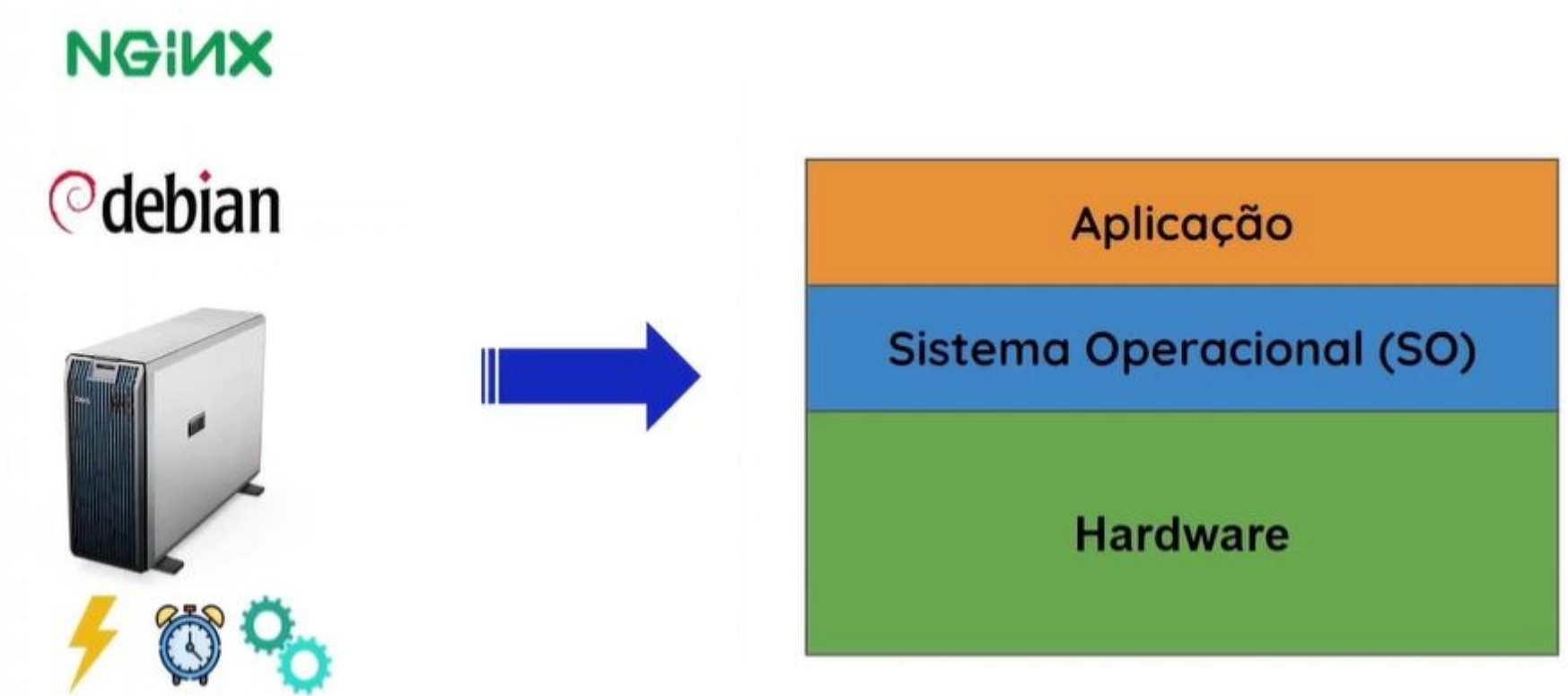
Como funcionava?

- **Muito tempo ocioso**
- **Recursos não aproveitados**

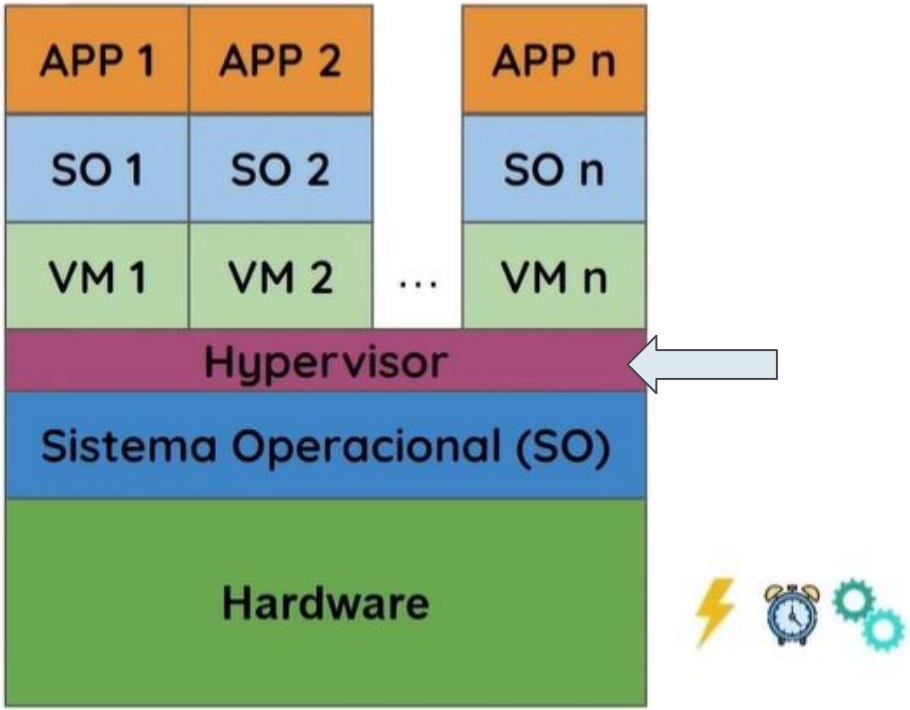
Como funcionava?

- Muito tempo ocioso
- Recursos não aproveitados
- Alto preço de manutenção

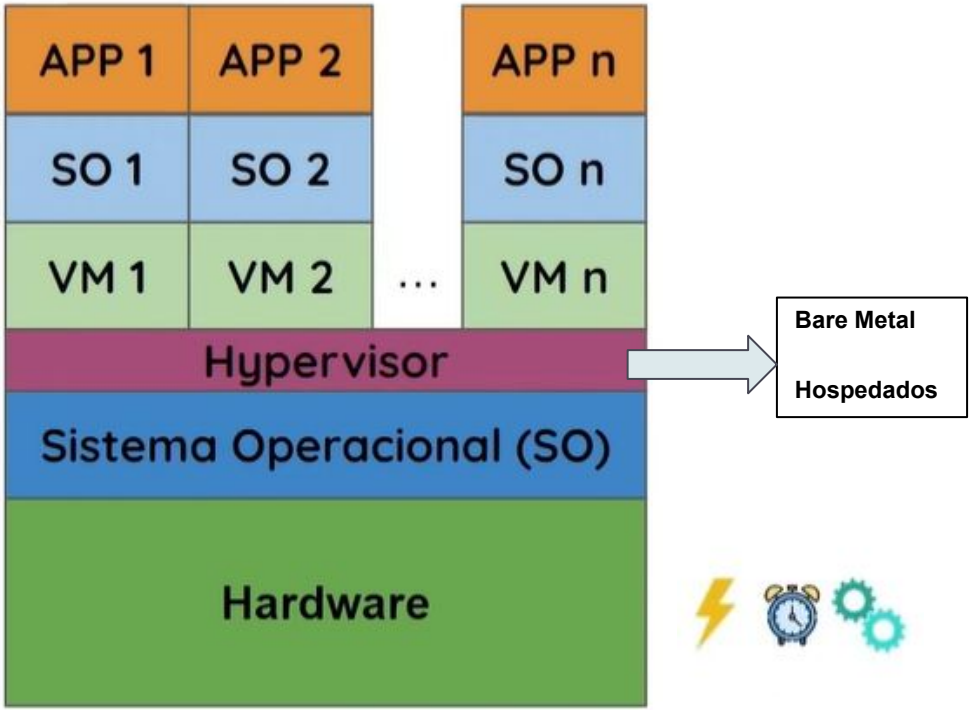
Como funcionava?



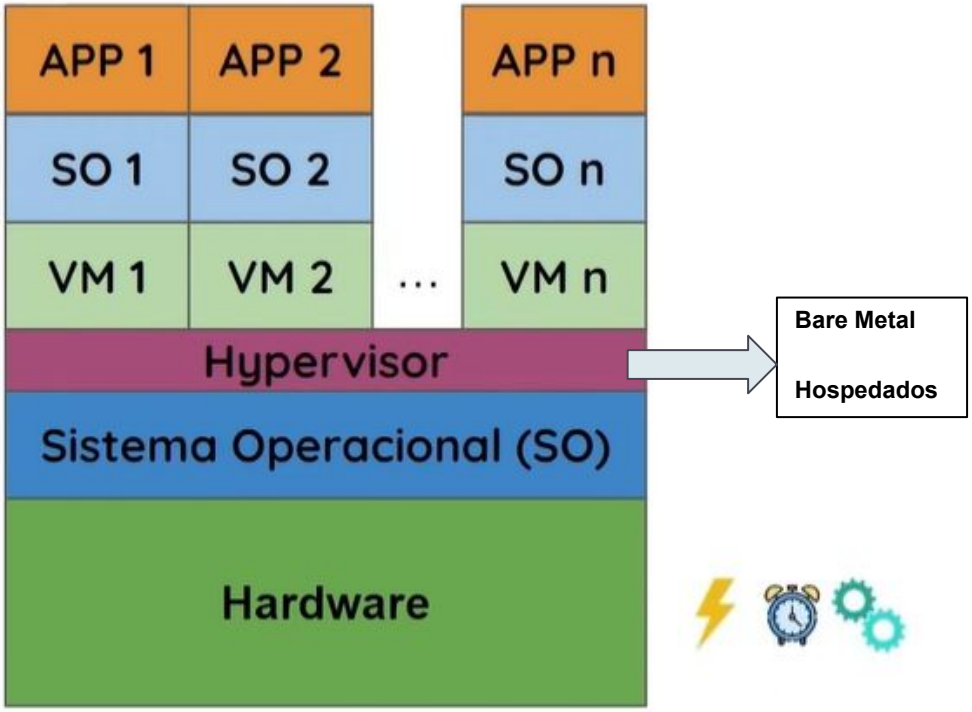
Máquinas Virtuais (VMs)



Máquinas Virtuais (VMs)

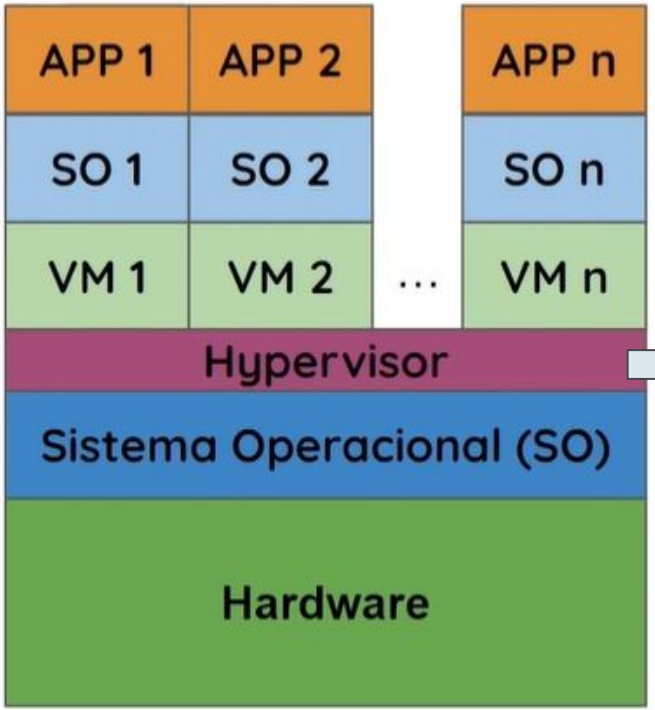


Máquinas Virtuais (VMs)



- **Vantagens**
  - Capacidade melhor aproveitada
  - Pouco tempo ocioso
  - Redução dos custos físicos
- **E a manutenção?**

Máquinas Virtuais (VMs)



1GB de RAM

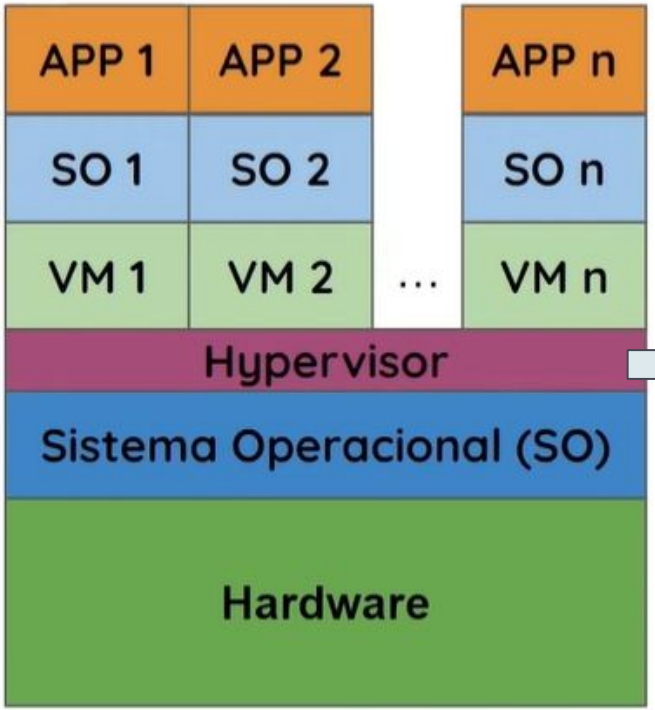
10GB de Disco

10% de Processamento

Bare Metal

Hospedados

Máquinas Virtuais (VMs)



1GB de RAM

10GB de Disco

10% de Processamento

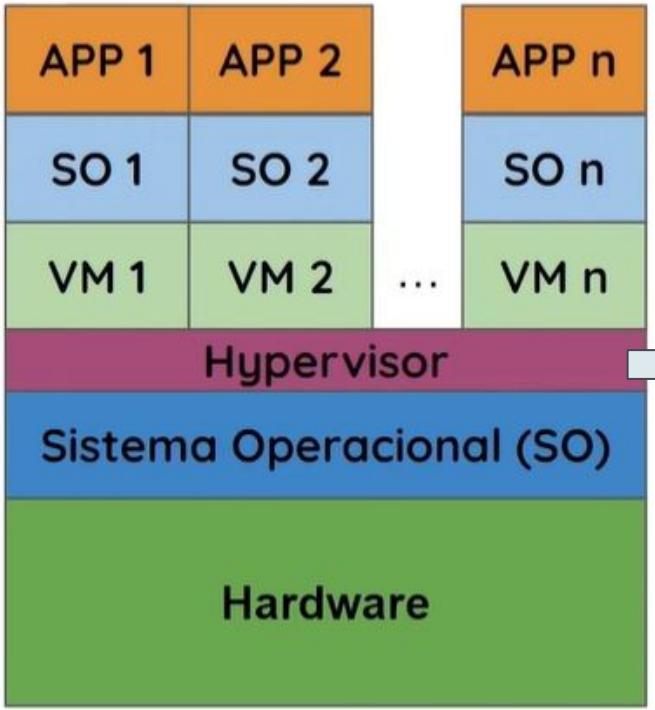
4 VMs

4x

Bare Metal

Hospedados

Máquinas Virtuais (VMs)



1GB de RAM

10GB de Disco

10% de Processamento

4 VMs

4x

4GB de RAM

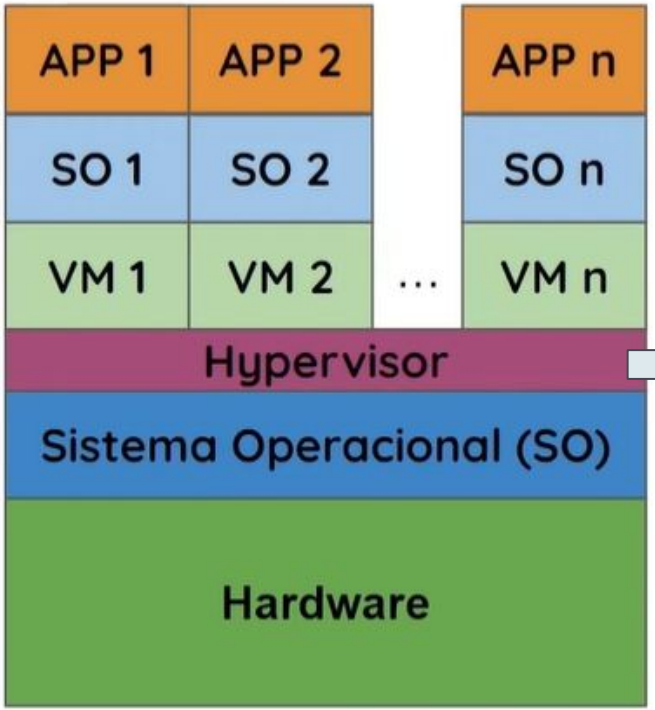
40GB de Disco

40% de Processamento

Bare Metal

Hospedados

Máquinas Virtuais (VMs)



1GB de RAM

10GB de Disco

10% de Processamento

4 VMs

4x

4GB de RAM

40GB de Disco

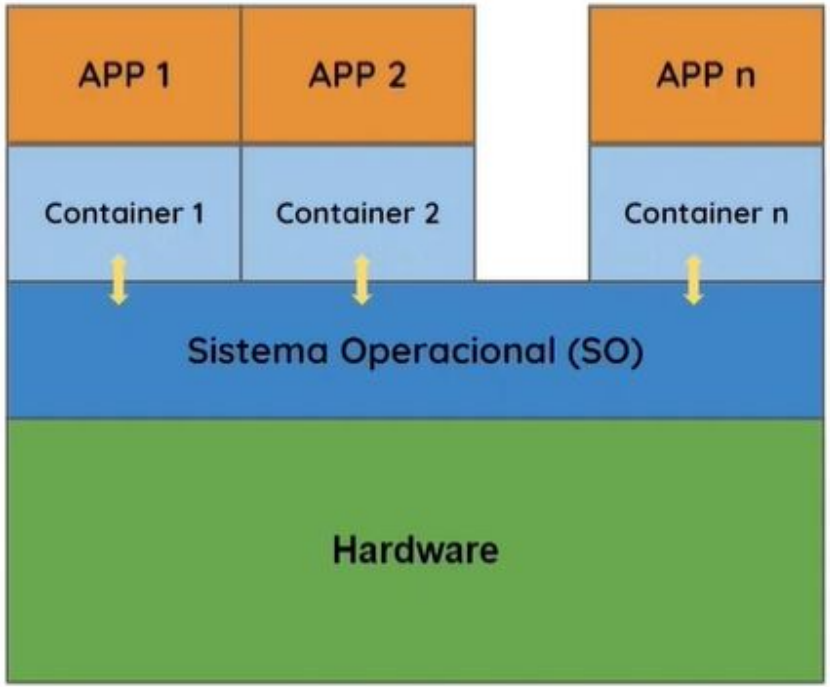
40% de Processamento

Bare Metal

Hospedados

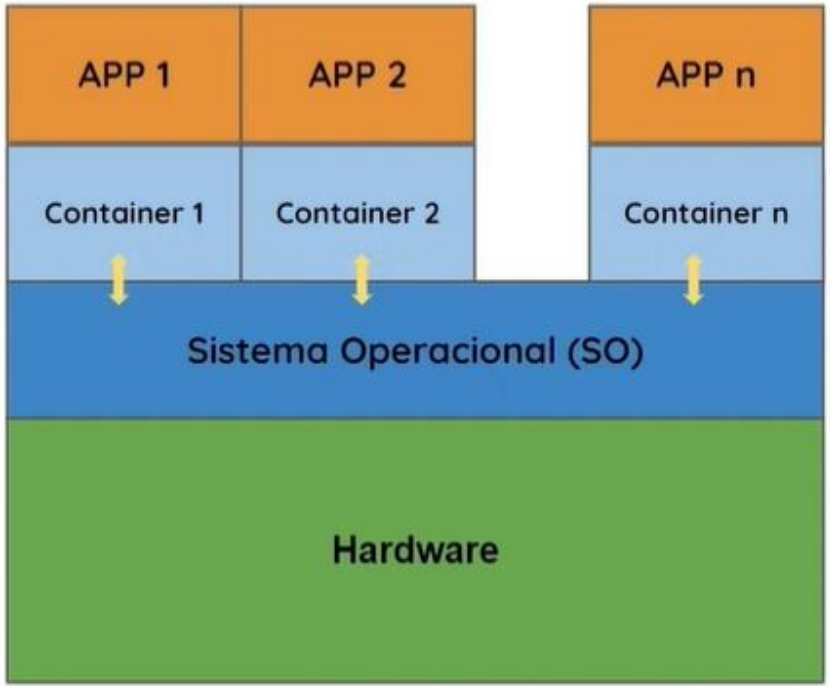
- **Todos os custos de manutenção continuariam**
  - Configuração
  - Atualização
  - Segurança
  - Tempo

Containers



- **Muito mais leve**
- **Nascem/morrem rápido**
- **Sem custo de manutenção do Sistema Operacional**

Containers

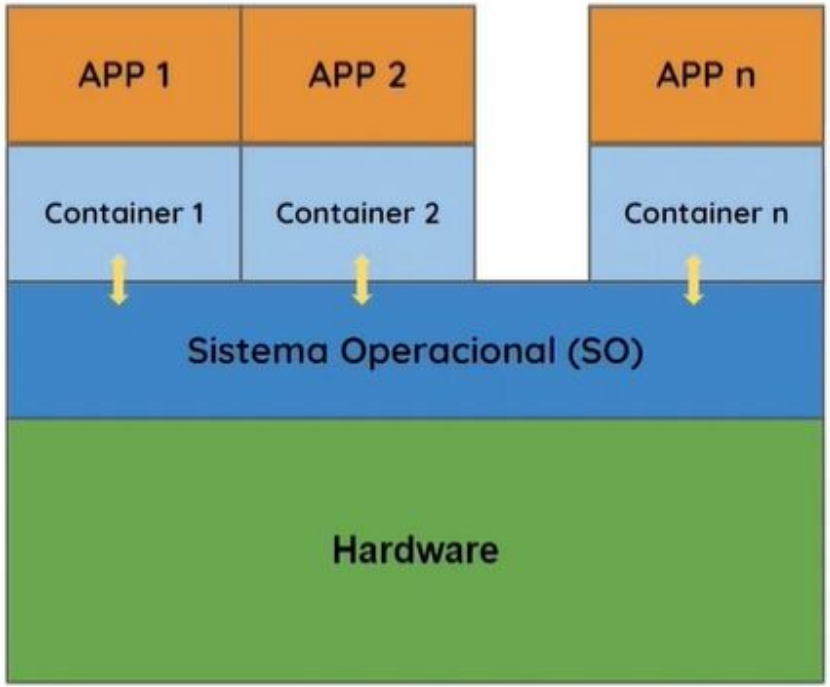


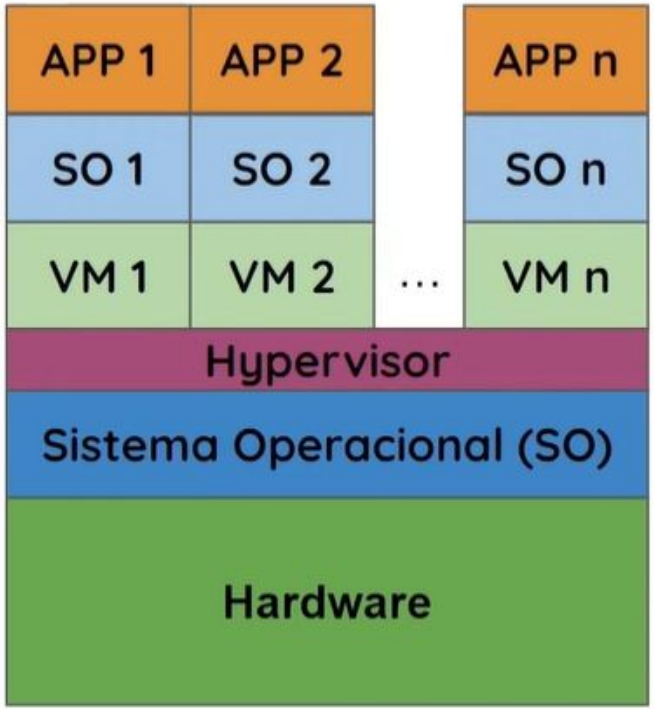
1. Muito mais leve

2. Nascem/morrem rápido
   - 2.1. Efêmero
     - 2.1.1. Que é passageiro, temporário, transitório.

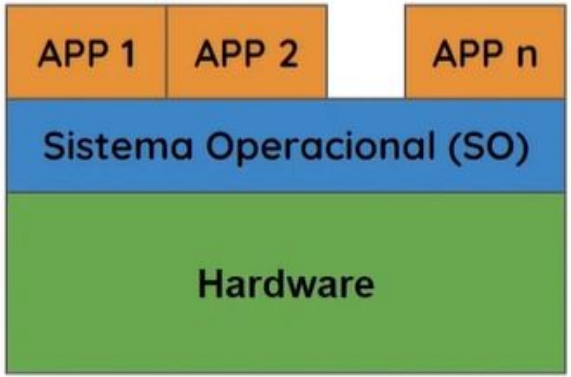
3. Sem custo de manutenção do Sistema Operacional

Containers


APP 1
APP 2
APP n
Container 1
Container 2
Container n
Sistema Operacional (SO)
Hardware


APP 1
APP 2
APP n
SO 1
SO 2
SO n
VM 1
VM 2
...
VM n
Hypervisor
Sistema Operacional (SO)
Hardware

E se não usarmos Containers?



- **E se uma aplicação consumir muito recurso?**

E se não usarmos Containers?

- **E se uma aplicação consumir muito recurso?**
- **E se cada aplicação precisa de uma versão específica de uma biblioteca?**
- **E se mais de uma aplicação precisar utilizar a mesma porta?**

Utilizando Containers?

- Ganhamos Isolamento;
- Controle de recursos;
- Agilidade na infraestrutura;
- Possibilidade de trabalhar com diversas versões das bibliotecas;
- Muito mais leve do que VM;
- Não requer manutenção com SO's.

Histórico do Docker

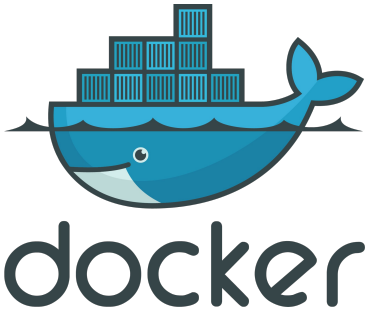
Histórico do Docker

Solomon Hykes

Histórico do Docker

Solomon Hykes

Histórico do Docker



Solomon Hykes

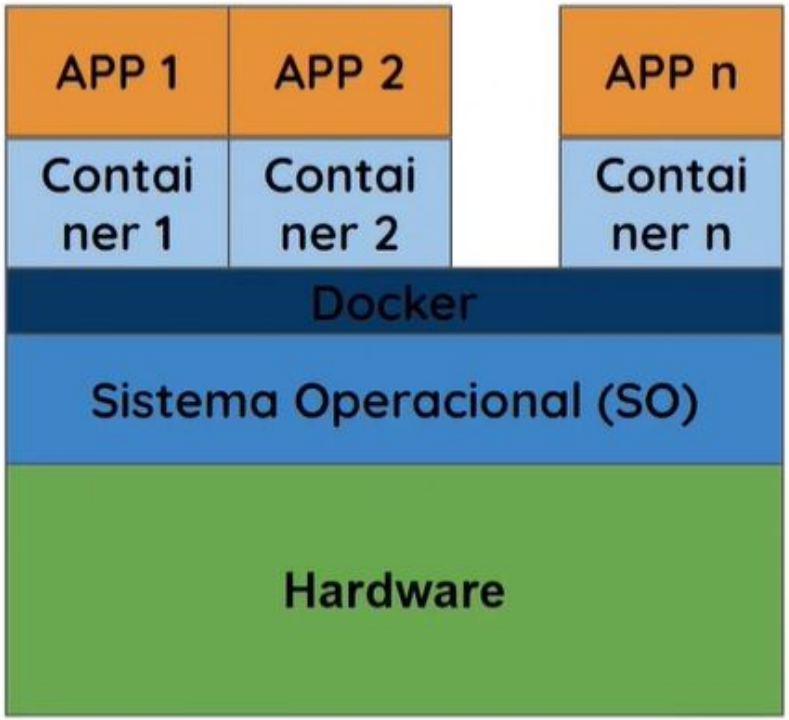
Histórico do Docker

Solomon Hykes

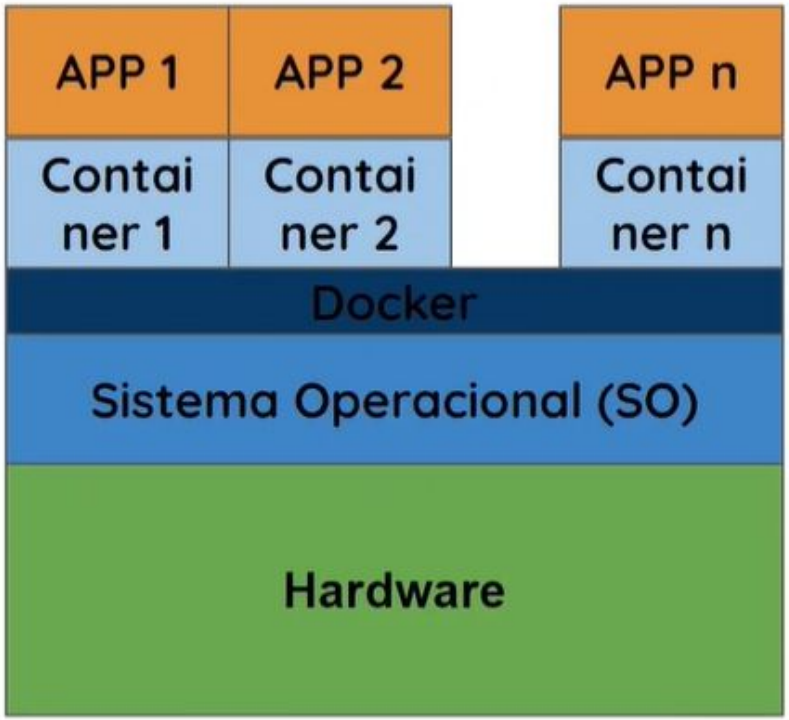
Open Source

- https://github.com/docker

Doker

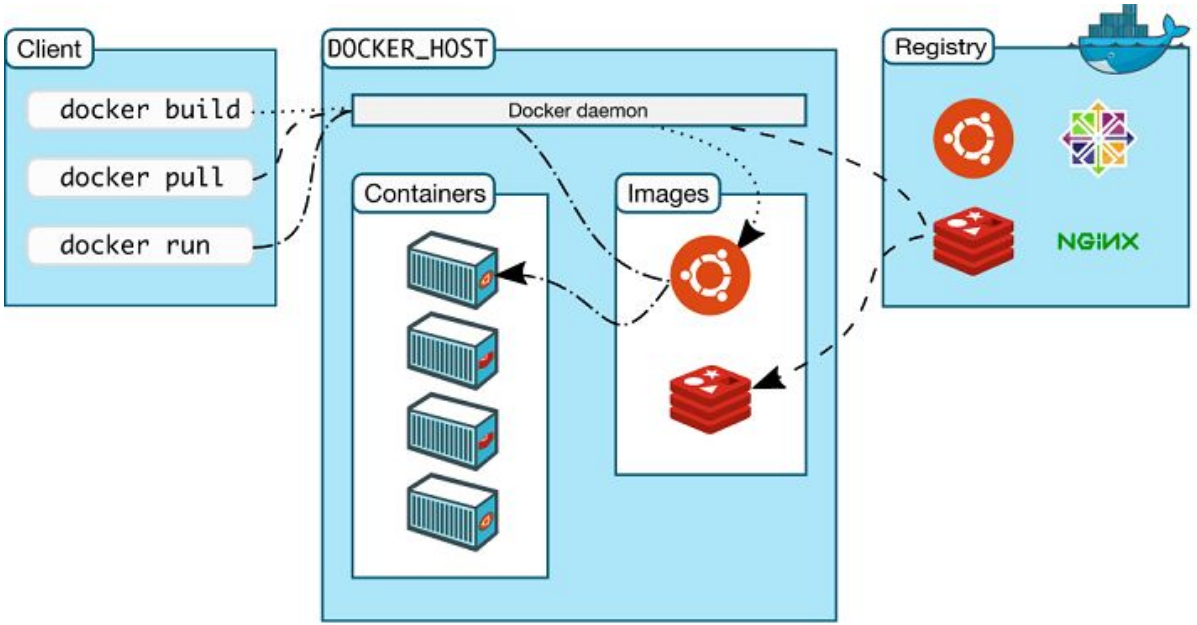

APP 1
APP 2
APP n
Contai ner 1
Contai ner 2
Contai ner n
Docker
Sistema Operacional (SO)
Hardware

Doker



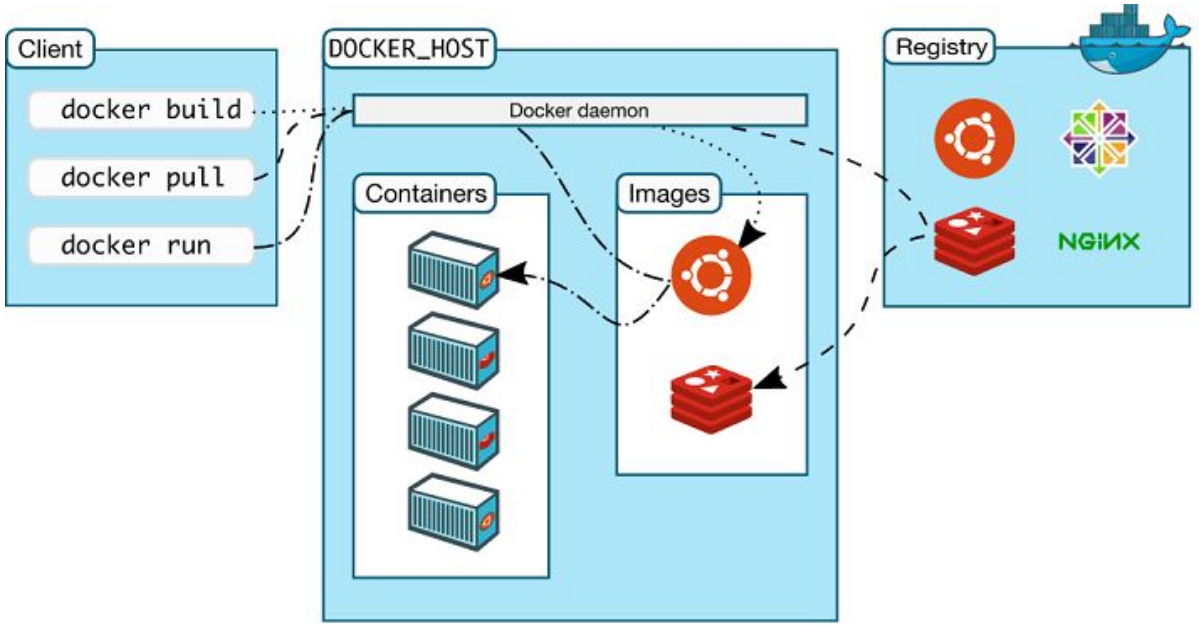
Docker Engine

Docker Arquitetura



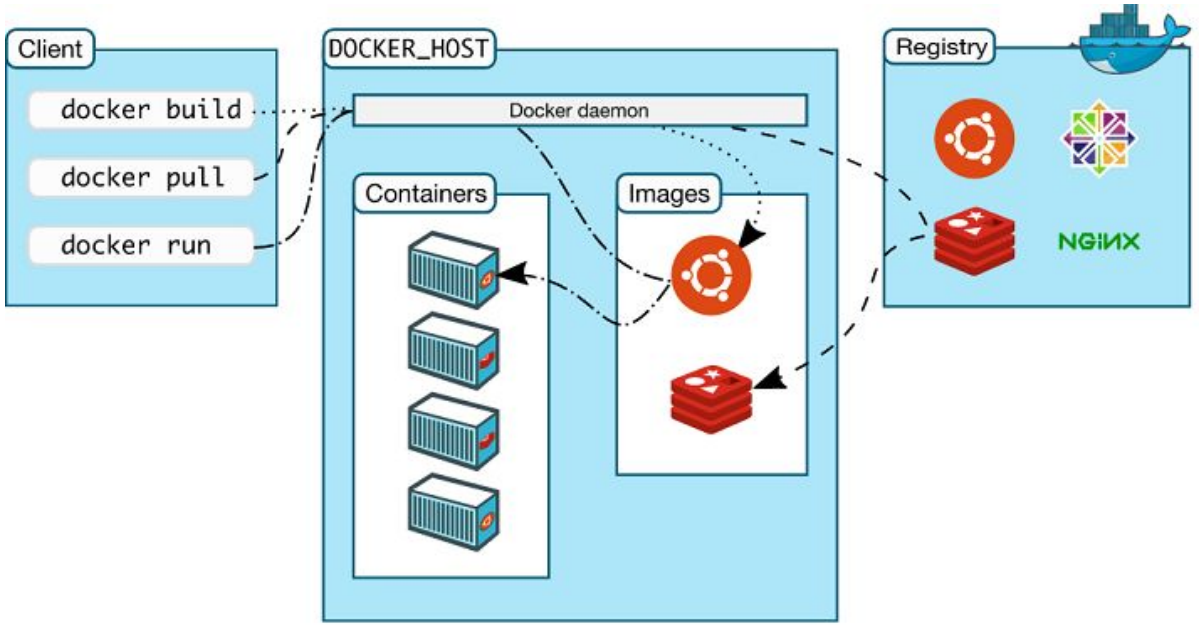
- **Cliente**
  - Onde iremos digitar os comandos
- **Docker_Host**
  - O Docker possui uma arquitetura de cliente-servidor, e tais contêineres são armazenados em um servidor, chamado de Docker host ou Docker server
- **Registry**
  - É uma espécie de repositório para imagens.

Docker Arquitetura



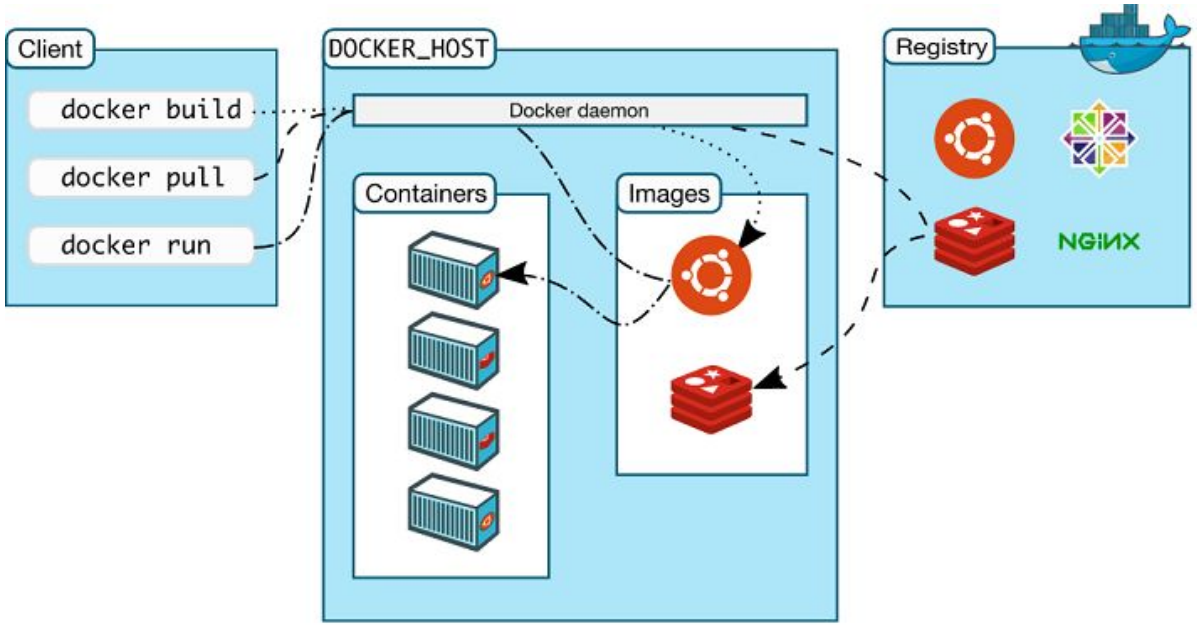
- **Docker daemon**
  - Recebe comandos do cliente a partir de Command Line Interfaces ou API's REST
  - Interpreta os comandos

Docker Arquitetura



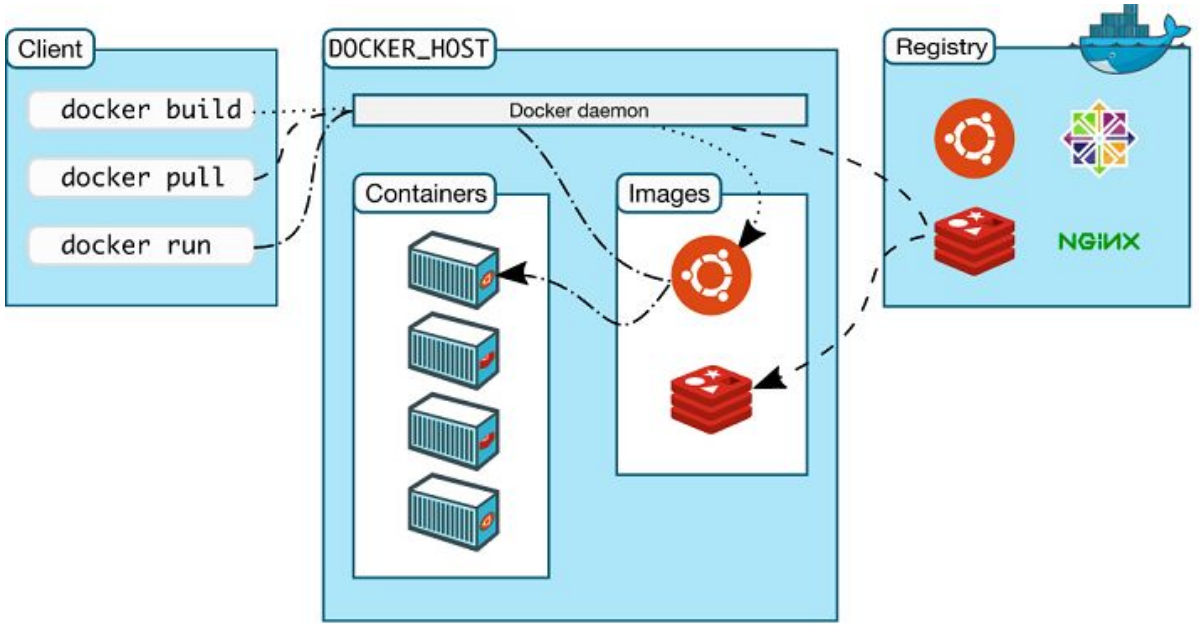
- **Registry**
  - Caso não exista a imagem o Docker daemon irá conversar com o **Registry** e vai baixar a imagem

Docker Arquitetura



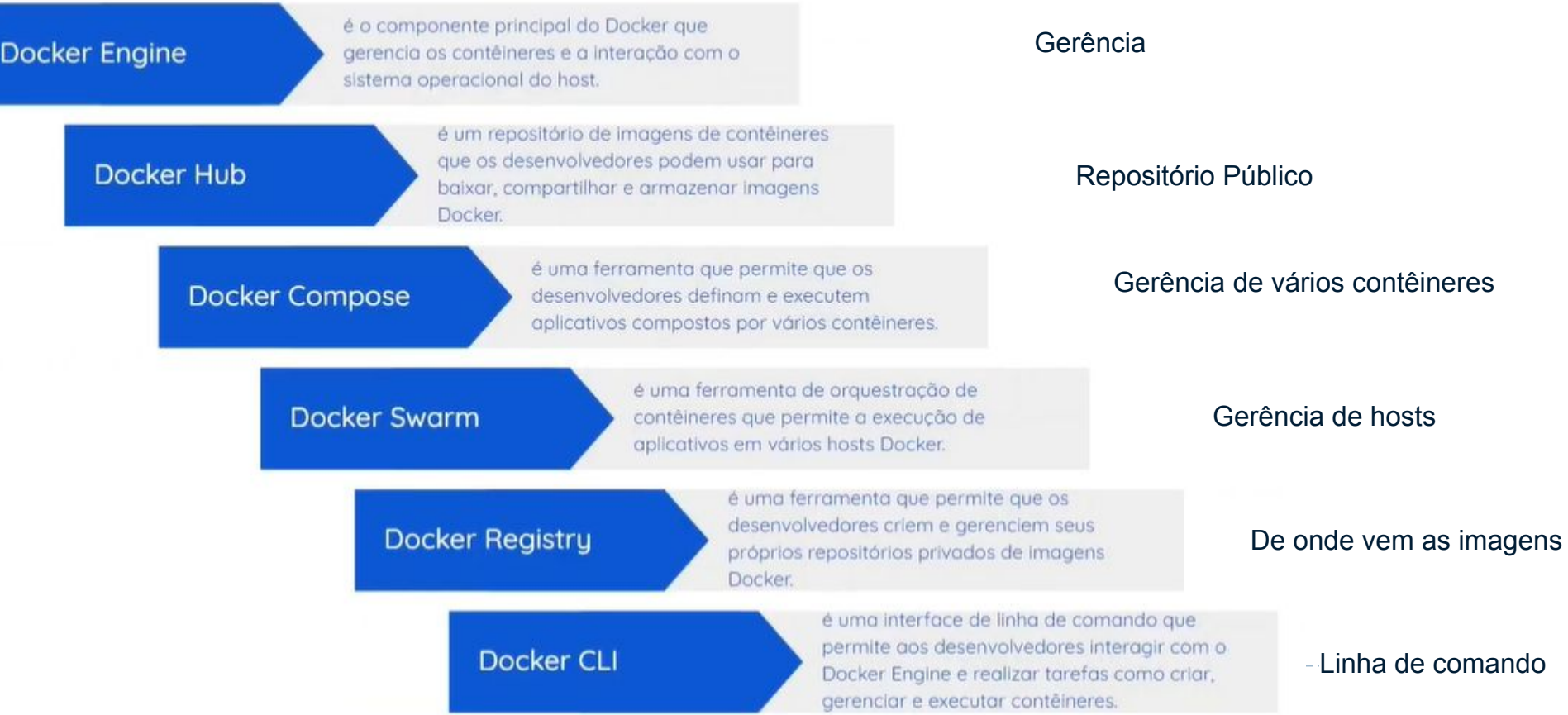
- **Imagens**
  - São arquivos que contém todo o conteúdo e estrutura de sistemas operacionais. Elas são a base de construção de contêineres no Docker.

Docker Arquitetura



- **Containers**
  - São os ambientes de execução do Docker, criados a partir de imagens. De forma simplificada, é uma sandbox para processos.

Tecnologias do Ecossistema Docker



Install Docker Engine on Ubuntu

- **Execute o seguinte comando para desinstalar todos os pacotes conflitantes:**
- for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove $pkg; done
- **Instalar usando o apt repositório:**
- sudo apt-get update
- sudo apt-get install ca-certificates curl
- sudo install -m 0755 -d /etc/apt/keyrings
- sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
- sudo chmod a+r /etc/apt/keyrings/docker.asc
- **Adicionar ao repositório:**
- echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
- sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
- sudo apt-get update
- **Para instalar a versão mais recente, execute:**
- sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
- **Verifique se a instalação do Docker Engine foi bem-sucedida executando a hello-world imagem:**
- sudo service docker start
- sudo docker run hello-world

Docker comandos

- `sudo service docker status`
- `sudo service docker start`
- `sudo service docker stop`
- Inicializar quando iniciar a máquina:
  - `sudo systemctl enable docker.service`
  - `sudo systemctl enable containerd.service`
- Parar a inicialização
  - `sudo systemctl disable docker.service`
  - `sudo systemctl disable containerd.service`
- `docker version`

Docker comandos

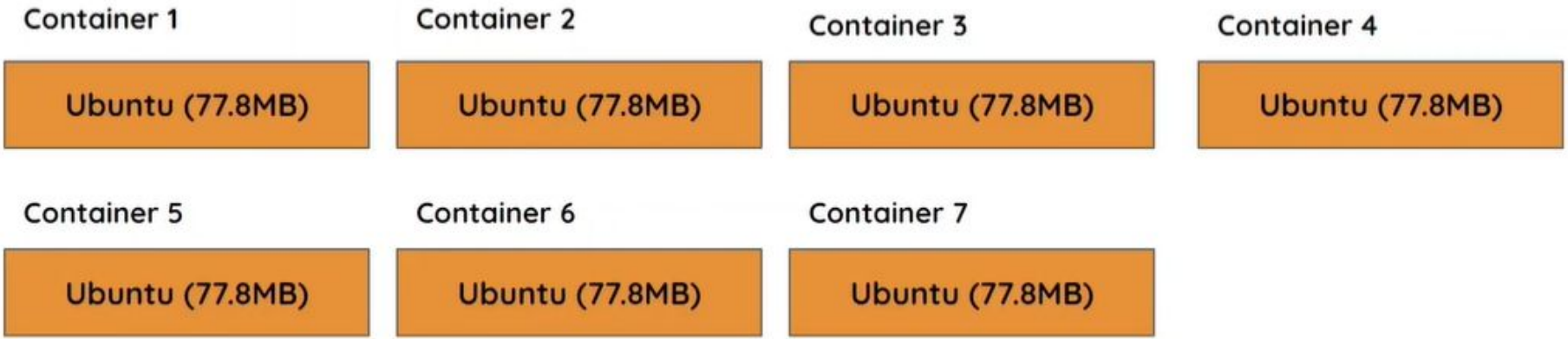
- verificar os docker ativos:
- docker ps
- docker container ls -a

Docker

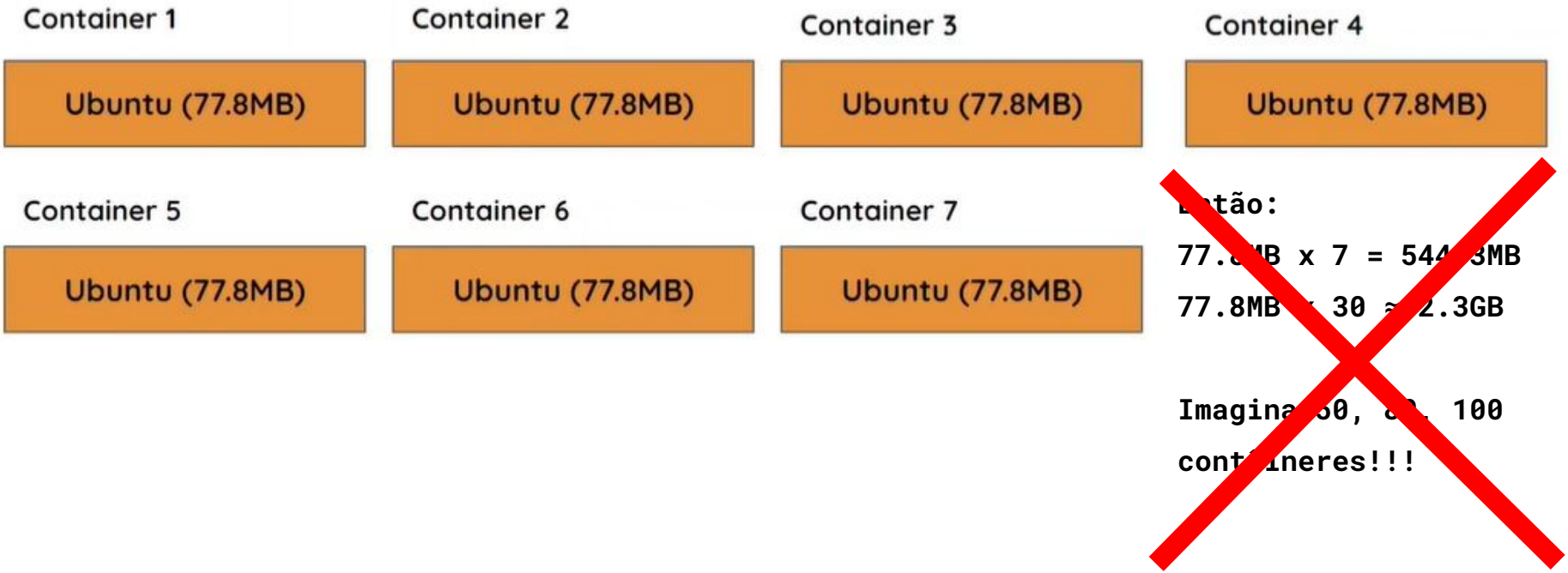
- **Instalação no Windows**
- https://learn.microsoft.com/pt-br/windows/wsl/install
- **Instalação Ubuntu**
- https://docs.docker.com/engine/install/ubuntu/
- https://docs.docker.com/engine/install/linux-postinstall/
- **Outra alternativa de usar o Docker**
- https://hub.docker.com/
- https://labs.play-with-docker.com/

Virtualização - Ubuntu 22.04

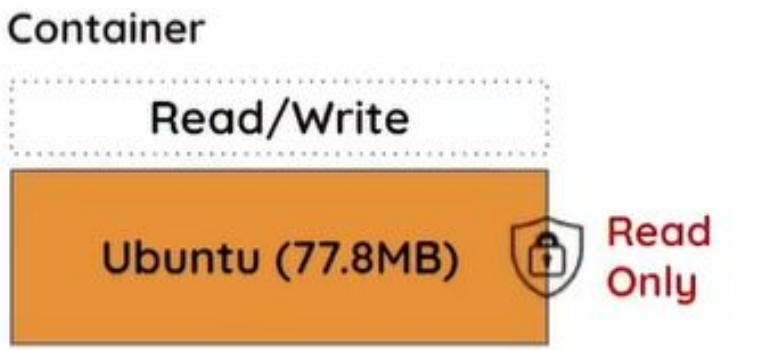
Seria assim?


Container 1
Ubuntu (77.8MB)
Container 2
Ubuntu (77.8MB)
Container 3
Ubuntu (77.8MB)
Container 4
Ubuntu (77.8MB)
Container 5
Ubuntu (77.8MB)
Container 6
Ubuntu (77.8MB)
Container 7
Ubuntu (77.8MB)

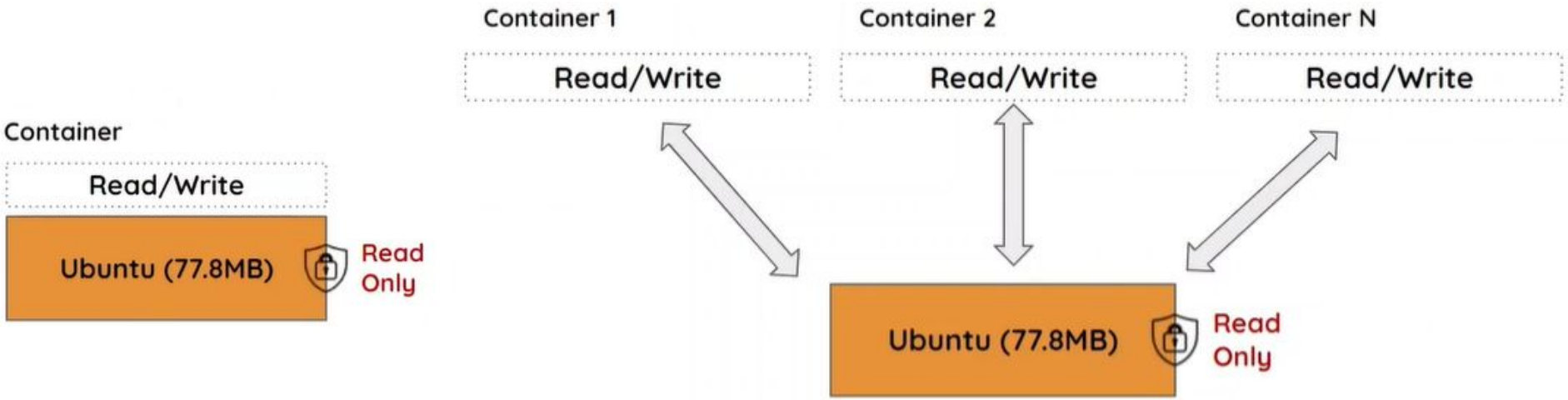
Seria assim?


Container 1
Ubuntu (77.8MB)
Container 2
Ubuntu (77.8MB)
Container 3
Ubuntu (77.8MB)
Container 4
Ubuntu (77.8MB)
Container 5
Ubuntu (77.8MB)
Container 6
Ubuntu (77.8MB)
Container 7
Ubuntu (77.8MB)
Então:
77.8MB x 7 = 544.3MB
77.8MB x 30 = 2.3GB
Imagina 50, 100
containeres!!!

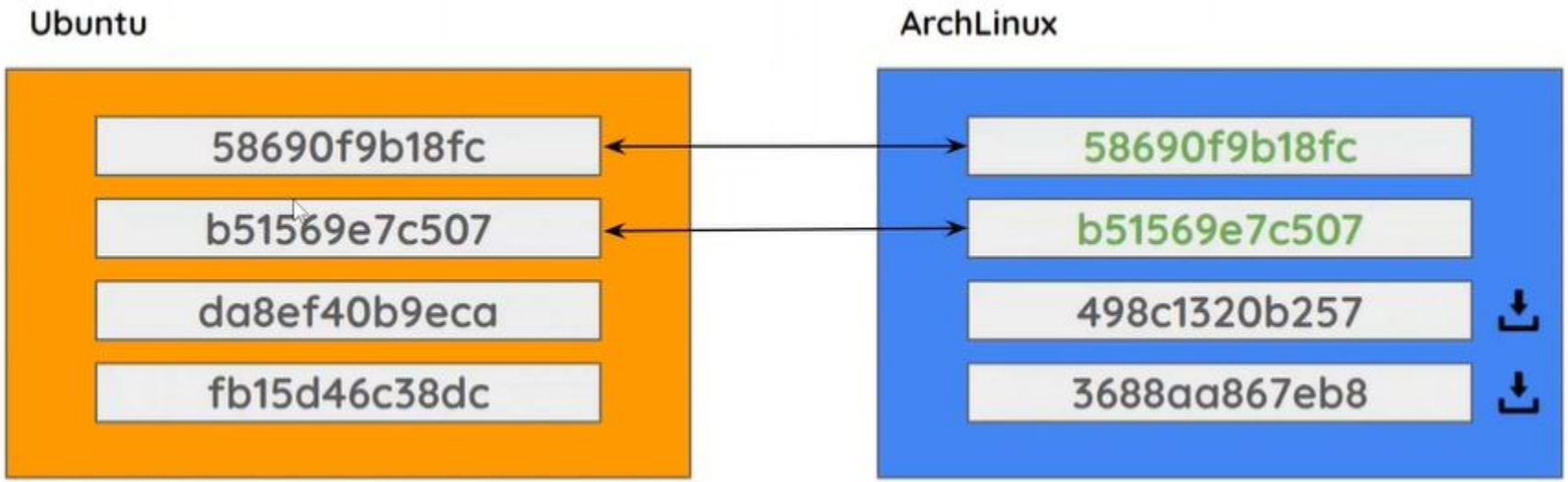
Estrutura dos containers

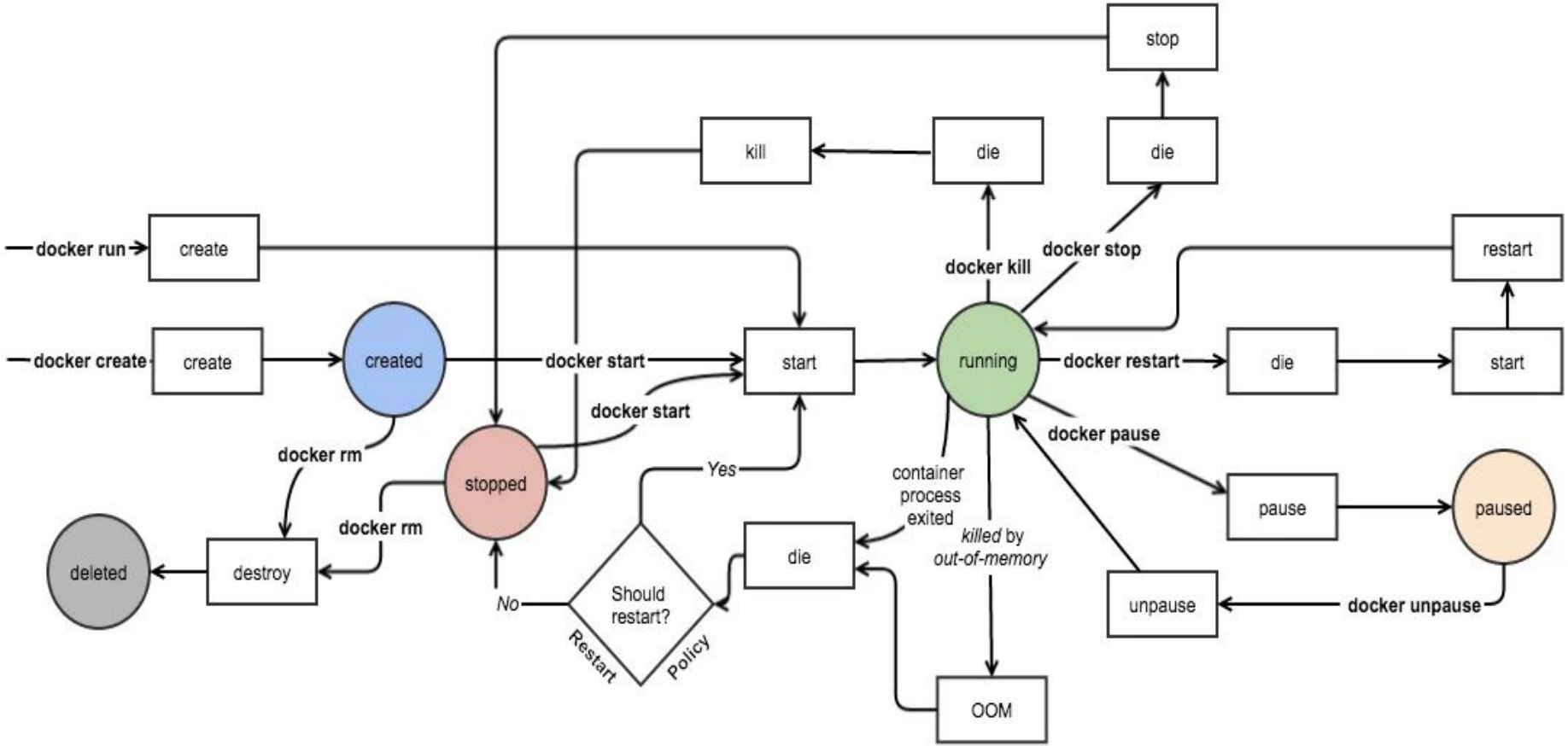


Estrutura dos containers



Aproveitamento de camadas




stop
kill
die
die
docker run
create
docker kill
docker stop
restart
docker create
create
created
docker start
start
running
docker restart
die
start
docker start
docker rm
docker pause
stopped
container process exited
Yes
pause
paused
killed by out-of-memory
docker rm
deleted
destroy
die
Should restart?
No
Restart Policy
unpause
docker unpause
OOM

Comandos

docker create --name <container-name> <image-name>

docker run -it -d --name <container-name> <image-name> bash

docker pause <container-id/name>

docker unpause <container-id/name>

docker start <container-id/name>

docker stop <container-id/name>

docker restart <container-id/name>

docker kill <container-id/name>

docker rm <container-id/name>

Comandos

```
docker run hello-world
docker run ubuntu
docker run --name teste1 -i -t ubuntu
docker exec id echo “GUTO”
docker run ubuntu echo "eu gosto muito"
docker run -dit ubuntu
docker rm -f ID
docker rmi  ID
docker inspect nginx:latest
docker run -p 5000:80 -d -e AUTHOR="GUTO"  dockersamples/static-site
```

UniSenac Centro Universitário RS

Senac Fecomércio Sesc

MUITO OBRIGADO!!!!!

Guto Muniz

augustomuniz@gmail.com